|  |  |
| --- | --- |
| EOL  End Of Life | |
| **Summary** | This is the Software Detailed Design Document for the *DAIMLER MMA* Project. |

|  |  |  |
| --- | --- | --- |
| **Author** | **Review** | **Approval** |
| Title: Stefan Dominte | See Project Master Document for the roles and Project Members List for the name of people | See Project Master Document for the roles and Project Members List for the name of people |
|  |  |  |
| **Distribution** | | |
| See Project Master Document for the roles and Project Members List for the name of people | See Project Master Document for the roles and Project Members List for the name of people | See Project Master Document for the roles and Project Members List for the name of people |

Table of contents

[General Information 4](#_Toc134434249)

[1.1. Revision history \* 4](#_Toc134434250)

[1.2. Purpose and Scope 4](#_Toc134434251)

[1.3. Referenced documents 4](#_Toc134434254)

[1.3.1. External documents 4](#_Toc134434255)

[1.3.2. Internal Documents 4](#_Toc134434256)

[1.3.3. Terminology and definitions 5](#_Toc134434257)

[2. SW Module Detailed Design 6](#_Toc134434258)

[2.1. Overview 6](#_Toc134434259)

[2.2. Traceability 6](#_Toc134434260)

[3. Features 8](#_Toc134434261)

[3.1. Services 8](#_Toc134434262)

[3.1.1. EOL\_runMainFunction 8](#_Toc134434263)

[3.1.2. eol\_ GetCounterIndex 9](#_Toc134434596)

[3.1.3. eol\_ManageCycleCounters 10](#_Toc134434597)

[3.1.4. eol\_CalculateObsolescenseData 11](#_Toc134434598)

[3.1.5. EOL\_Get\_Obsolescence\_Data 12](#_Toc134434599)

[3.1.6. EOL\_Write\_Obsolescence\_Data 13](#_Toc134434600)

[3.1.7. eol\_GetVariableCounterValue 14](#_Toc134434601)

[3.2. Runnabes – Auto-tests 15](#_Toc134434602)

[3.2.1. EOL\_Autotest\_CheckProductEndOfLifeComfort 15](#_Toc134434603)

[3.2.2. EOL\_Autotest\_CheckProductEndOfLifeLowForce 16](#_Toc134434604)

[3.2.3. EOL\_Autotest\_CheckProductEndOfLifeHighForce 17](#_Toc134434605)

[3.2.4. EOL\_Autotest\_CheckProductEndOfLifeMaxForce 18](#_Toc134434606)

[3.3. Variabiles 19](#_Toc134434607)

[3.3.1. EOL\_u8OldExecutedCycleNumber 19](#_Toc134434608)

[3.3.2. eol\_stObsolescenseData 19](#_Toc134434609)

[3.4. Macros 20](#_Toc134434610)

[3.4.1. KU8\_EOL\_COMFORT\_LEVEL 20](#_Toc134434611)

[3.4.2. KU8\_EOL\_LOW\_FORCE\_LEVEL 20](#_Toc134434612)

[3.4.3. KU8\_EOL\_HIGH\_FORCE\_LEVEL 20](#_Toc134434613)

[3.4.4. KU8\_EOL\_MAX\_FORCE\_LEVEL 20](#_Toc134434614)

[3.4.5. KU8\_EOL\_REQUEST\_COUNTER\_LEVEL 20](#_Toc134434615)

[3.4.6. COUNTER\_LIMIT\_ENABLED 21](#_Toc134434616)

[3.4.7. KU8\_NB\_MONITORED\_CYCLES 21](#_Toc134434617)

[3.4.8. KU8\_VARIABLE\_COUNTER\_LOW 21](#_Toc134434618)

[3.4.9. KU8\_VARIABLE\_COUNTER\_HIGH 21](#_Toc134434619)

[3.4.10. KU8\_VARIABLE\_COUNTER\_MAX 21](#_Toc134434620)

[3.4.11. KU8\_VARIABLE\_COUNTER\_COMFORT 22](#_Toc134434621)

[4. EEPROM 22](#_Toc134434622)

[5. Configuration 22](#_Toc134434629)

[6. Compilation Options 22](#_Toc134434630)

# Tables of figures

[Figure 1: EOL\_runMainFunction 9](#_Toc134298179)

[Figure 2: eol\_GetCounterIndex Part1 10](#_Toc134298180)

[Figure 3: eol\_GetCounterIndex Part2 10](#_Toc134298181)

[Figure 4: eol\_ManageCycleCounters 11](#_Toc134298182)

[Figure 5: eol\_CalculateObsolescenseData: 12](#_Toc134298183)

[Figure 6: EOL\_Get\_Obsolescence\_Data 13](#_Toc134298184)

[Figure 7: EOL\_Write\_Obsolescence\_Data 14](#_Toc134298185)

[Figure 8: eol\_GetVariableCounterValue Part1 15](#_Toc134298186)

[Figure 9: eol\_GetVariableCounterValue Part2 15](#_Toc134298187)

[Figure 10: EOL\_Autotest\_CheckProductEndOfLifeComfort 16](#_Toc134298188)

[Figure 11: EOL\_Autotest\_CheckProductEndOfLifeLowForce 17](#_Toc134298189)

[Figure 12: EOL\_Autotest\_CheckProductEndOfLifeHighForce 18](#_Toc134298190)

[Figure 13: EOL\_Autotest\_CheckProductEndOfLifeMaxForce 19](#_Toc134298191)

# General Information

## Revision history \*

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author(s)** | **Description/comment** |
| 1.1.2.2 | 08/06/2022 | Stefan Dominte | Initial revision |
| 1.1.2.3 | 16/06/2022 | Stefan Dominte | Traceability updated |
| 1.1.2.4 | 16/06/2022 | Stefan Dominte | Traceability reupdated |
| 1.1.2.5 | 29/08/2022 | Stefan Dominte | Updated for 3.0 release |
| 1.1.2.6 | 15/11/2022 | Stefan Dominte | Updated for 4.0 release |
| 1.1.2.7 | 06/02/2023 | Stefan Dominte | Updated for 5.0 release |
| 1.1.2.8 | 05/05/2023 | Stefan Dominte | Updated for 6.0 release |
| 1.1.2.9 | 21/06/2023 | Madalina Serban | Updated for 6.1 release |
| 1.1.2.10 | 22/06/2023 | Madalina Serban | Updated traceability |
| 1.1.2.11 | 22/06/2023 | Madalina Serban | Updated internal document version |
| 1.1.2.12 | 22/06/2023 | Madalina Serban | Updated internal document version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*\* Template history is found in the CM tool used for templates*

## Purpose and Scope

The purpose of this document is to provide an overview of the EOL operation principle, and to present the implementation choices in terms of module and function splitting.

## Referenced documents

### External documents

|  |  |  |
| --- | --- | --- |
| **Id** | **Title** | **Reference** |
|  |  |  |
|  |  |  |
|  |  |  |

### Internal Documents

|  |  |  |
| --- | --- | --- |
| **Id** | **Title** | **Reference** |
|  | DAIMLER\_MMA\_SWarchitectureDesignInterfaceDescription.docx |  |
|  | EOL - Design Interface Description.docx |  |
|  | SBE\_4G\_NVP\_layout.xls |  |
|  |  |  |

### Terminology and definitions

|  |  |
| --- | --- |
| **Terminology** | **Meaning** |
| AAU | Atomic architectural unit |
| SW | Software |
|  |  |

# SW Module Detailed Design

## Overview

EOL (End Of Life) is a very small SW unit intended to count the number of times belt functions are triggered and to notify that the end of life of a belt function (or a group of belt functions) is reached.

From a static point of view, EOL is connected to 3 SW units:

* BFE: To read / detect any cycles start (based on executed cycle data)
* NVP: To load EOL thresholds, to record EOL counters
* CIL: To read / detect any PreSafe request and the specific level requested

## Traceability

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | Criteria | Linked Runnable | Source |
| DSG\_EOL\_0001 | This component shall update the low force counter when HWA profiles (cycles 18, 20, 22 and 24) or tensioning cycles 0, 1 are executed. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0002 |
| DSG\_EOL\_0002 | This component shall update the high force counter when tensioning cycles 2 and 3 are executed. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0001 |
| DSG\_EOL\_0003 | This component shall update the comfort counter when BSR profiles (cycles 26,27 and 28) or tensioning cycles 14, 15, 16 and 17 are executed. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0003 |
| DSG\_EOL\_0004 | The local variable EOL\_u8OldExecutedCycleNumber will hold the the old value of the executed cycle number | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0007 |
| DSG\_EOL\_0005 | The component shall update the Low force counter when the High force counter is incremented. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0008 |
| DSG\_EOL\_0006 | This component shall check if the end of life of the “low force” belt functions is reached. | EOL\_Autotest\_CheckProductEndOfLifeLowForce() | ARCH\_SW\_EOL\_0004 |
| DSG\_EOL\_0007 | This component shall check if the end of life of the “high force” belt functions is reached. | EOL\_Autotest\_CheckProductEndOfLifeHighForce() | ARCH\_SW\_EOL\_0005 |
| DSG\_EOL\_0008 | This component shall check if the end of life of the “comfort” belt functions is reached. | EOL\_Autotest\_CheckProductEndOfLifeComfort() | ARCH\_SW\_EOL\_0006 |
| DSG\_EOL\_0009 | This component shall read the executed cycle number in order to increment specific counter. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0009 |
| DSG\_EOL\_0010 | This component shall check if the end of life of the “max force” belt functions is reached. | EOL\_Autotest\_CheckProductEndOfLifeMaxForce() | ARCH\_SW\_EOL\_0012 |
| DSG\_EOL\_0011 | This component shall update the max force counter tensioning cycles 4, 5 and 6 are executed. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0010 |
| DSG\_EOL\_0012 | This component shall update the request counter when tensioning cycles 0-6 or HWA profile are executed. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0011 |
| DSG\_EOL\_0013 | The local function eol\_ManageCycleCounters will read the trigger request value in order to check if a PRE-SAFE request is active. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0013 |
| DSG\_EOL\_0014 | The local function eol\_ManageCycleCounters will read the pre safe level value in order to check which presafe level is active. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0014 |
| DSG\_EOL\_0015 | The local function eol\_ManageCycleCounters will update the NVM block status. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0015; ARCH\_SW\_EOL\_0023 |
| DSG\_EOL\_0016 | The local function eol\_ManageCycleCounters will update the Obsolescense Data block number. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0016 |
| DSG\_EOL\_0017 | This component shall update the associated cycle counter (low, high, max, comfort) when tensioning cycles 7, 8, 9, 10, 11, 12 and 13 are executed. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0017 |
| DSG\_EOL\_0018 | The local function eol\_GetCounterIndex will set the corresponding counter to low when the associated cycle counter configuration value is 1. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0018 |
| DSG\_EOL\_0019 | The local function eol\_GetCounterIndexwill set the corresponding counter to high when the associated cycle counter configuration value is 2. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0019 |
| DSG\_EOL\_0020 | The local function eol\_GetCounterIndex will set the corresponding counter to max when the associated cycle counter configuration value is 4. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0020 |
| DSG\_EOL\_0021 | The local function eol\_GetCounterIndex will set the corresponding counter to comfort when the associated cycle counter configuration value is 8. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0021 |
| DSG\_EOL\_0022 | The local function eol\_GetCounterIndex will set the corresponding counter to low when the associated cycle counter configuration value is different than 1,2,4 and 8. | EOL\_RunMainFunction() | ARCH\_SW\_EOL\_0022 |

# Features

## Services

### EOL\_runMainFunction

|  |  |  |
| --- | --- | --- |
| Object | | |
| Call the main function of the EOL module | | |
| **Prototype** | | |
| void EOL\_runMainFunction(void) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| OS | Every 10 ms | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

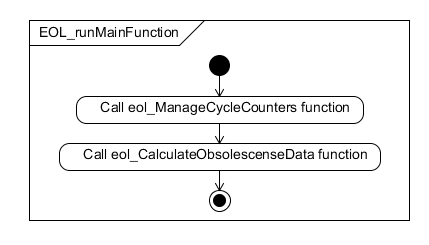


Figure : EOL\_runMainFunction

### eol\_ GetCounterIndex

|  |  |  |
| --- | --- | --- |
| Object | | |
| Defines the cycle counter | | |
| **Prototype** | | |
| LOCAL uint8 eol\_GetCounterIndex(const uint8 u8ExecutedCycleNumber) | | |
| **Input parameters** | | |
| Name | Type | Description |
| u8ExecutedCycleNumber | Uint8 | executed cycle number |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| EOL\_runMainFunction | Every main function call. | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

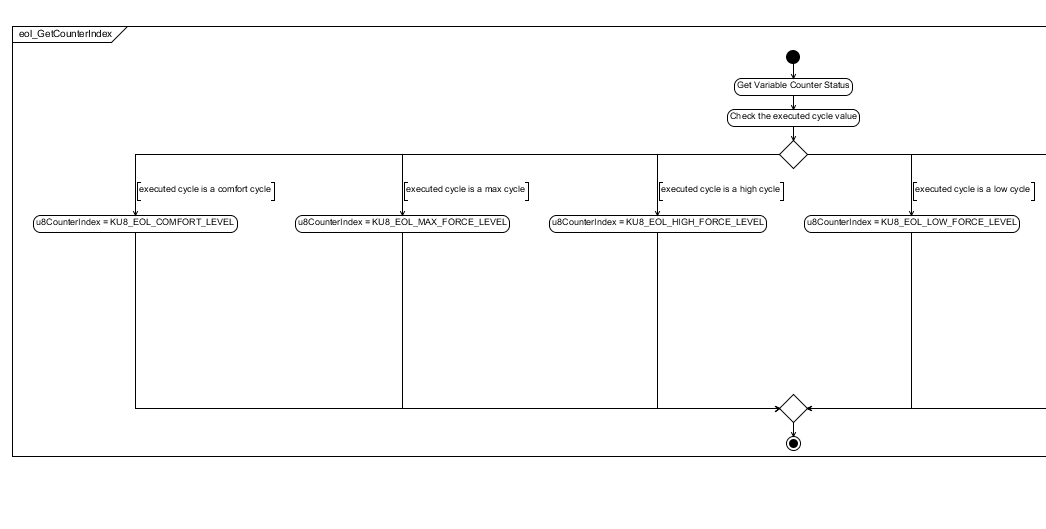


Figure : eol\_GetCounterIndex Part1

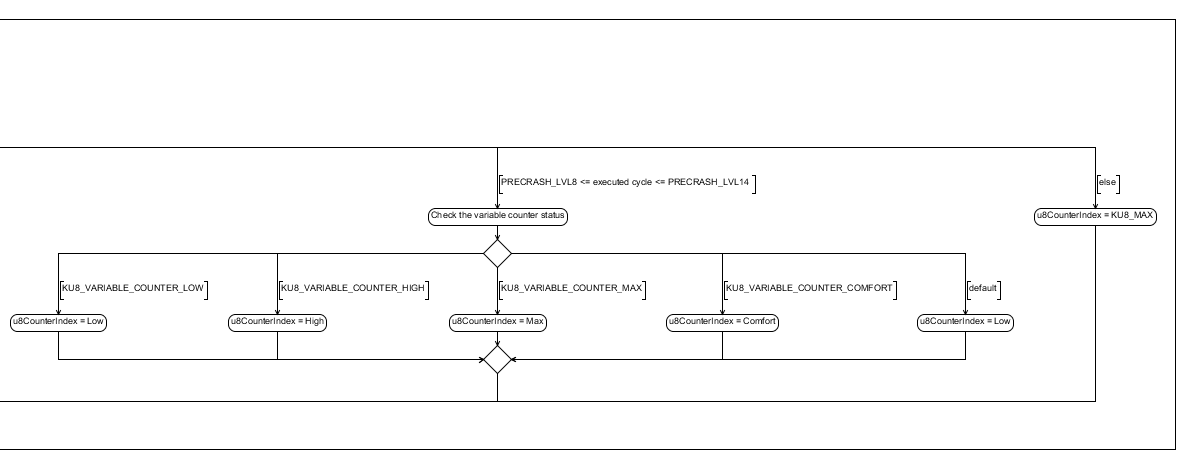


Figure : eol\_GetCounterIndex Part2

### eol\_ManageCycleCounters

|  |  |  |
| --- | --- | --- |
| Object | | |
| Computes cycles counters | | |
| **Prototype** | | |
| LOCAL uint8 eol\_GetCounterIndex(const uint8 u8ExecutedCycleNumber) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| EOL\_runMainFunction | Every main function call. | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

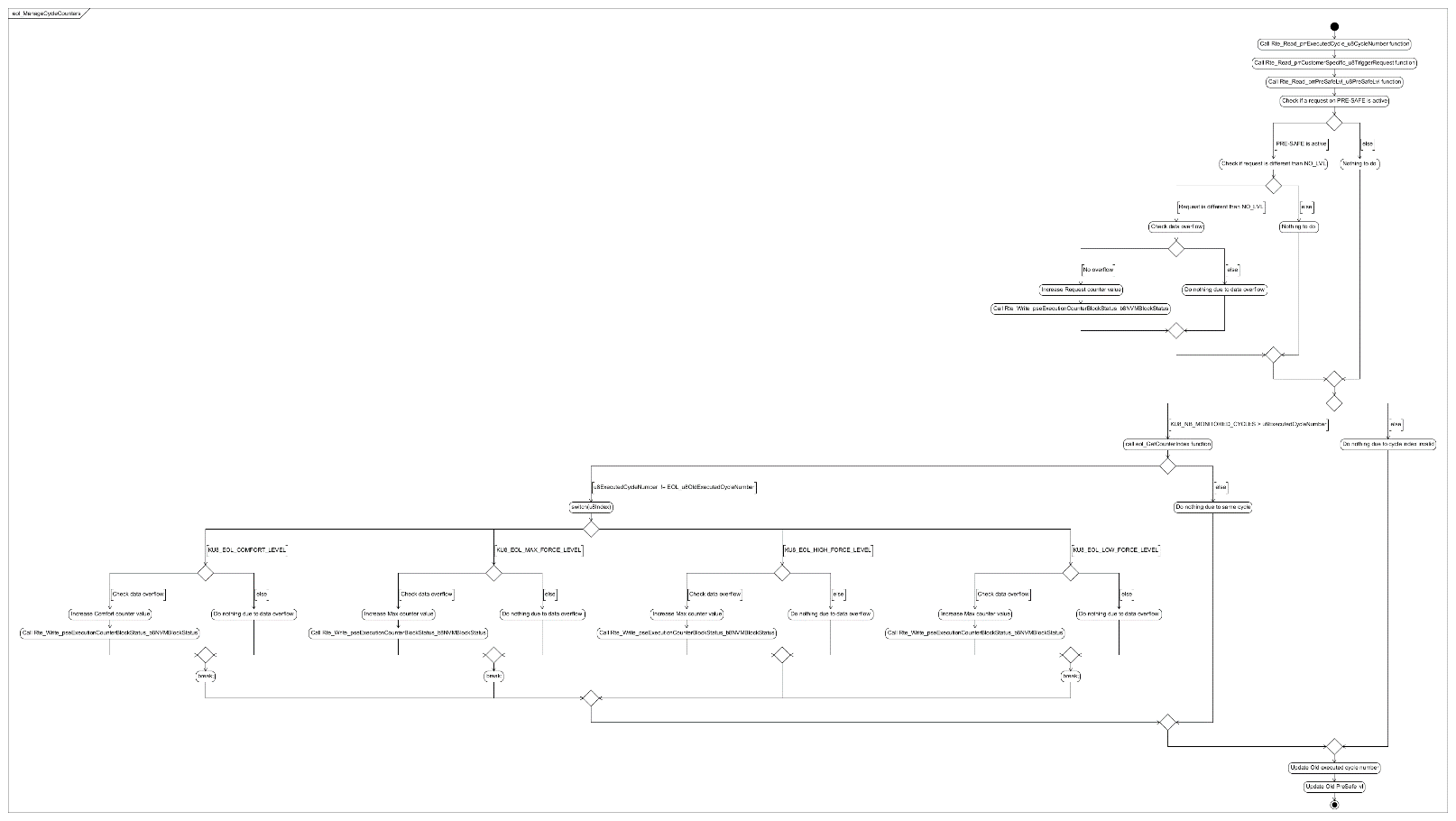


Figure : eol\_ManageCycleCounters

### eol\_CalculateObsolescenseData

|  |  |  |
| --- | --- | --- |
| Object | | |
| Computes obsolescense data values | | |
| **Prototype** | | |
| LOCAL void eol\_CalculateObsolescenseData(void) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| EOL\_runMainFunction | Every main function call. | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

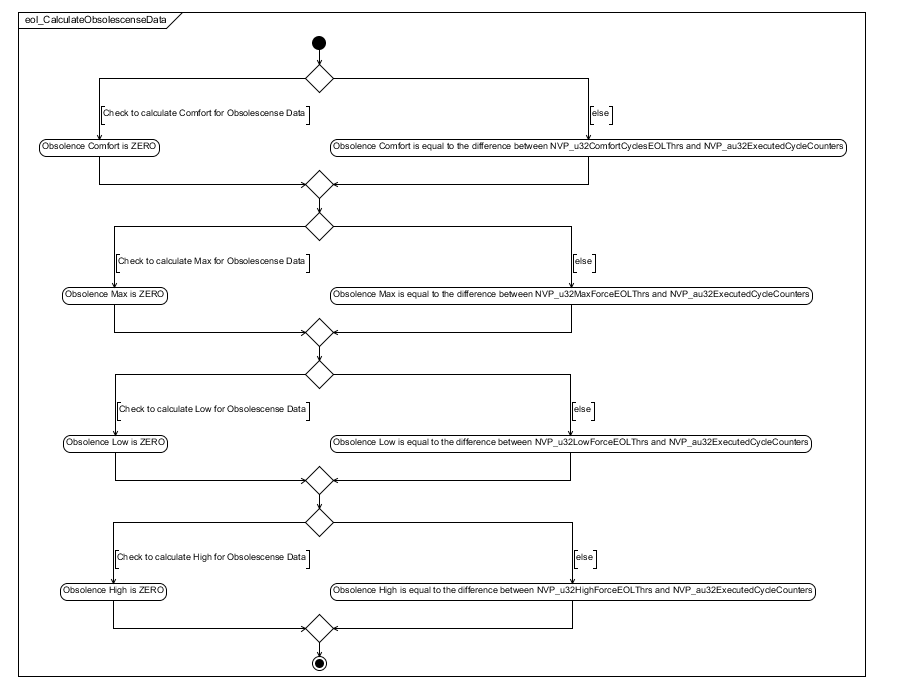


Figure : eol\_CalculateObsolescenseData:

### EOL\_Get\_Obsolescence\_Data

|  |  |  |
| --- | --- | --- |
| Object | | |
| Functions to get obsolence data | | |
| **Prototype** | | |
| FUNC(void, EOL\_AC\_EndOfLifeManagement\_CODE) EOL\_Get\_Obsolescence\_Data(P2CONST(Impl\_Obsolescense\_Data, AUTOMATIC, RTE\_EOL\_AC\_ENDOFLIFEMANAGEMENT\_APPL\_DATA) pObsData) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| pObsData | Impl\_Obsolescense\_Data | Obsolescense data |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
|  |  | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

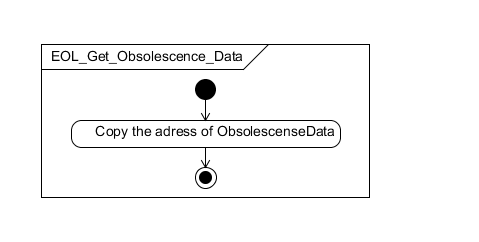


Figure : EOL\_Get\_Obsolescence\_Data

### EOL\_Write\_Obsolescence\_Data

|  |  |  |
| --- | --- | --- |
| Object | | |
| Functions to write obsolence data | | |
| **Prototype** | | |
| FUNC(void, EOL\_AC\_EndOfLifeManagement\_CODE) EOL\_Write\_Obsolescence\_Data(P2VAR(Impl\_Obsolescense\_Data, AUTOMATIC, RTE\_EOL\_AC\_ENDOFLIFEMANAGEMENT\_APPL\_VAR) pObsData) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| pObsData | Impl\_Obsolescense\_Data | Obsolescense data |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
|  |  | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

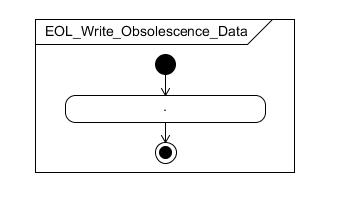


Figure : EOL\_Write\_Obsolescence\_Data

### eol\_GetVariableCounterValue

|  |  |  |
| --- | --- | --- |
| Object | | |
| Checks if a cycle has variable counter | | |
| **Prototype** | | |
| LOCAL uint8 eol\_GetVariableCounterValue(const uint8 u8ExecutedCycleNumber) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| u8ExecutedCycleNumber | uint8 | executed cycle number |
| **Return value** | | |
| Type | Description | |
| Uint8 | value of the variable counter or 0xFF if no counter is configured | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| eol\_GetCounterIndex |  | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

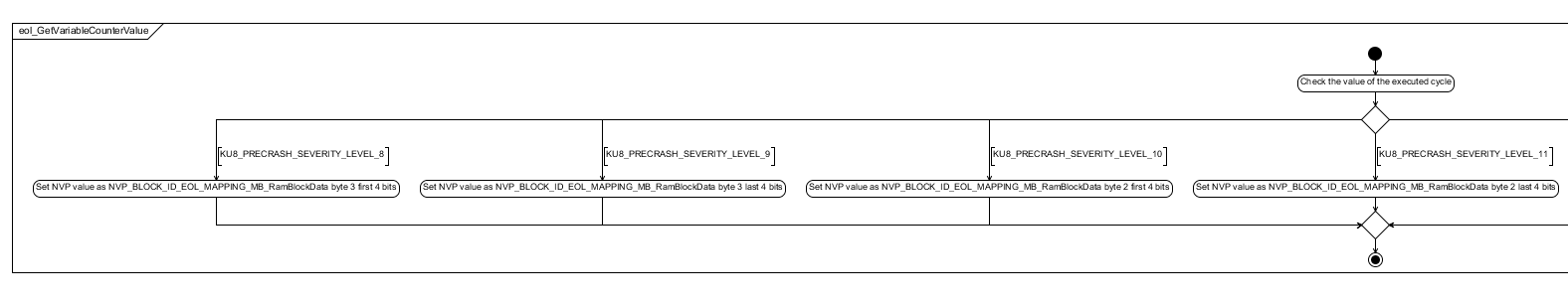


Figure : eol\_GetVariableCounterValue Part1

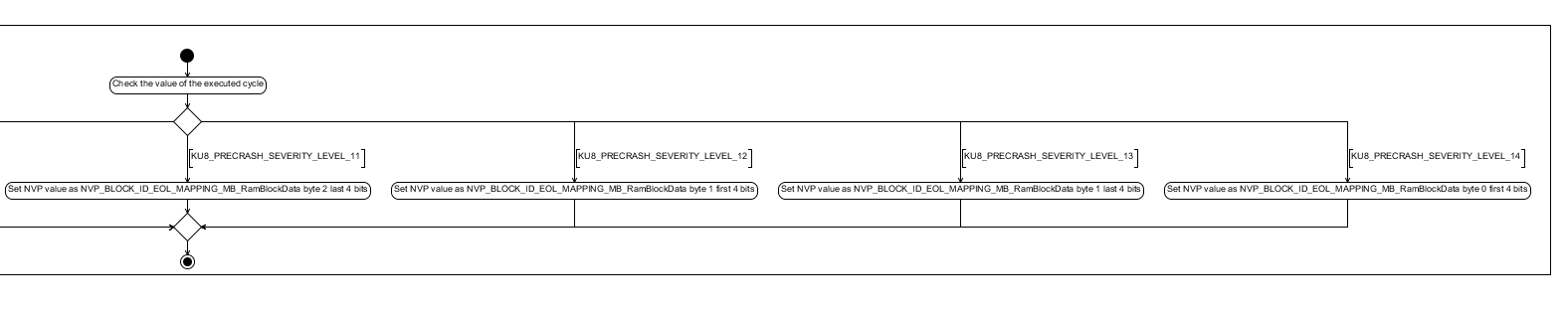


Figure : eol\_GetVariableCounterValue Part2

## Runnabes – Auto-tests

### EOL\_Autotest\_CheckProductEndOfLifeComfort

|  |  |  |
| --- | --- | --- |
| Object | | |
| Check if comfort counter "end of life" is reached or not | | |
| **Prototype** | | |
| void EOL\_Autotest\_CheckProductEndOfLifeComfort(u8TestResultType \* pu8TestResult) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| pu8TestResult | u8TestResultType | test result of the autotest |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| NA | NA | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

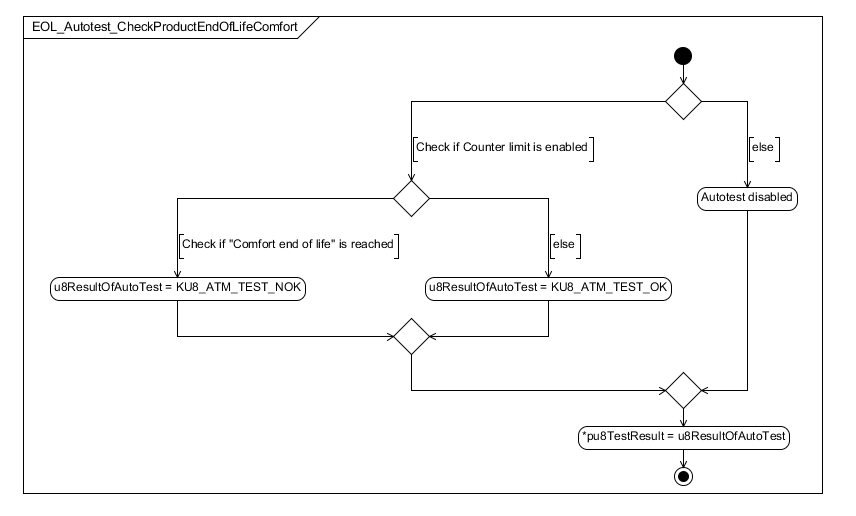


Figure : EOL\_Autotest\_CheckProductEndOfLifeComfort

### EOL\_Autotest\_CheckProductEndOfLifeLowForce

|  |  |  |
| --- | --- | --- |
| Object | | |
| Check if low force counter "end of life" is reached or not | | |
| **Prototype** | | |
| void EOL\_Autotest\_CheckProductEndOfLifeLowForce (u8TestResultType \* pu8TestResult) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| pu8TestResult | u8TestResultType | test result of the autotest |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| NA | NA | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

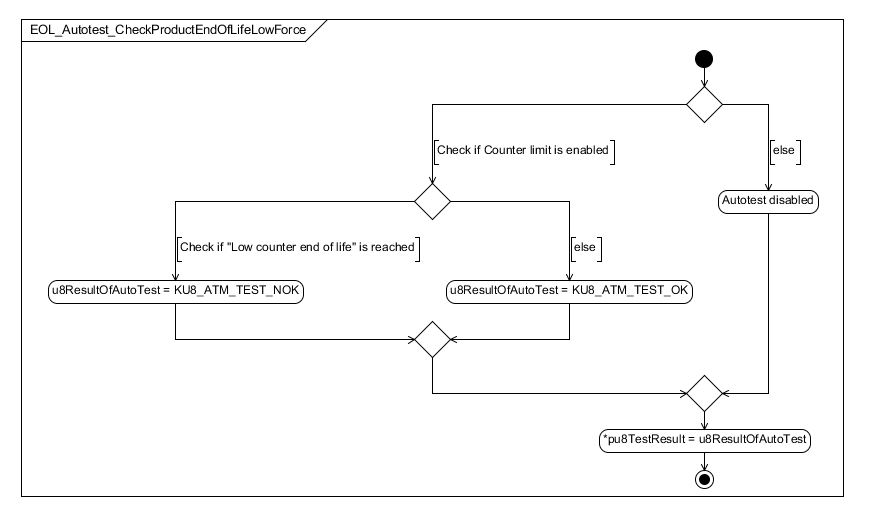


Figure : EOL\_Autotest\_CheckProductEndOfLifeLowForce

### EOL\_Autotest\_CheckProductEndOfLifeHighForce

|  |  |  |
| --- | --- | --- |
| Object | | |
| Check if high force counter "end of life" is reached or not | | |
| **Prototype** | | |
| void EOL\_Autotest\_CheckProductEndOfLifeHighForce (u8TestResultType \* pu8TestResult) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| pu8TestResult | u8TestResultType | test result of the autotest |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| NA | NA | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

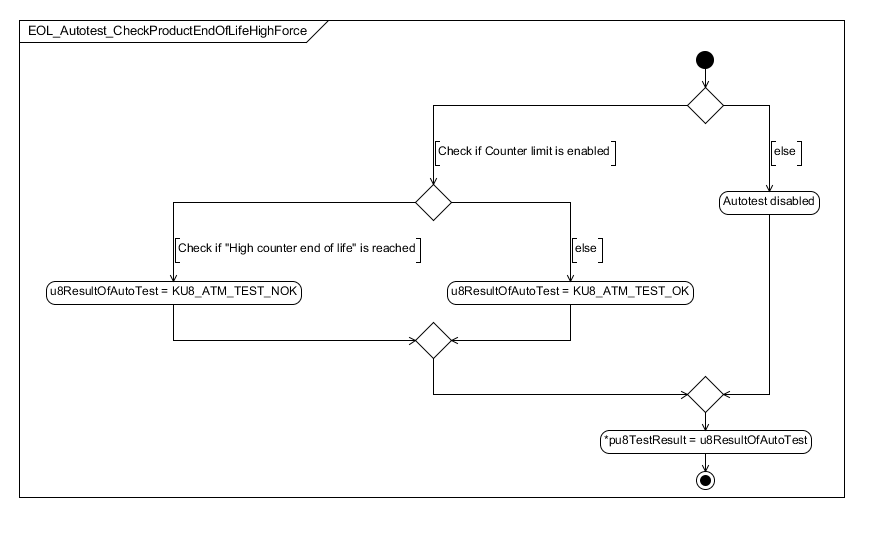


Figure : EOL\_Autotest\_CheckProductEndOfLifeHighForce

### EOL\_Autotest\_CheckProductEndOfLifeMaxForce

|  |  |  |
| --- | --- | --- |
| Object | | |
| Check if max force counter "end of life" is reached or not | | |
| **Prototype** | | |
| void EOL\_Autotest\_CheckProductEndOfLifeMaxForce (u8TestResultType \* pu8TestResult) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| pu8TestResult | u8TestResultType | test result of the autotest |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| NA | NA | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

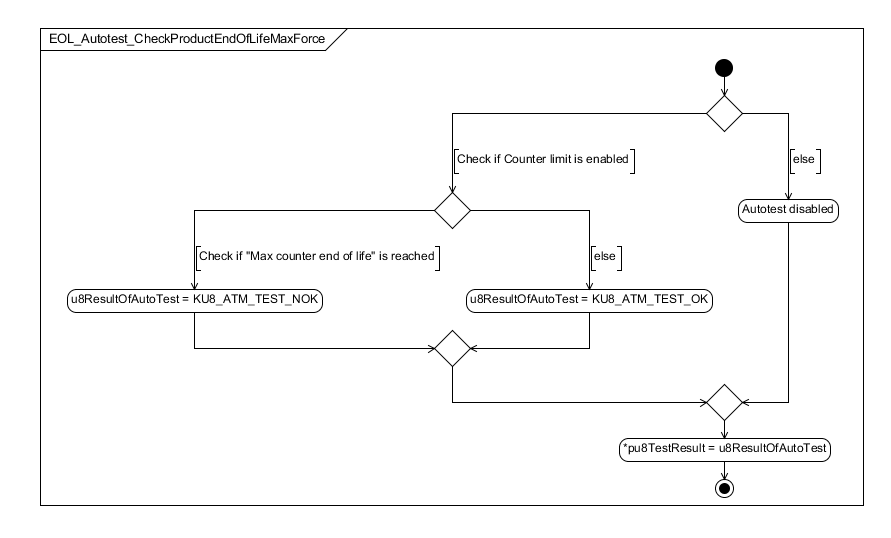


Figure : EOL\_Autotest\_CheckProductEndOfLifeMaxForce

## Variabiles

### EOL\_u8OldExecutedCycleNumber

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| u8CycleNumberType | 0xFF | |
| **Description** | | |
| Store the old value of the executed cycle number | | |
| **Definition** | | |
| LOCAL u8CycleNumberType EOL\_u8OldExecutedCycleNumber | | |

### eol\_stObsolescenseData

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| Impl\_Obsolescense\_Data | 0x00 | |
| **Description** | | |
| Store the obsolenscense data | | |
| **Definition** | | |
| LOCAL Impl\_Obsolescense\_Data eol\_stObsolescenseData = {0}; | | |

## Macros

### KU8\_EOL\_COMFORT\_LEVEL

|  |  |
| --- | --- |
| Name | Value |
| Comfort cycle counter ID | 0x00 |
| **Definition** | |
| #define KU8\_EOL\_COMFORT\_LEVEL | |
| **Description** | |
| NA | |

### KU8\_EOL\_LOW\_FORCE\_LEVEL

|  |  |
| --- | --- |
| Name | Value |
| Low force cycle counter ID | 0x01 |
| **Definition** | |
| #define KU8\_EOL\_LOW\_FORCE\_LEVEL | |
| **Description** | |
| NA | |

### KU8\_EOL\_HIGH\_FORCE\_LEVEL

|  |  |
| --- | --- |
| Name | Value |
| High force cycle counter ID | 0x02 |
| **Definition** | |
| #define KU8\_EOL\_HIGH\_FORCE\_LEVEL | |
| **Description** | |
| NA | |

### KU8\_EOL\_MAX\_FORCE\_LEVEL

|  |  |
| --- | --- |
| Name | Value |
| Max force cycle counter ID | 0x03 |
| **Definition** | |
| #define KU8\_EOL\_MAX\_FORCE\_LEVEL | |
| **Description** | |
| NA | |

### KU8\_EOL\_REQUEST\_COUNTER\_LEVEL

|  |  |
| --- | --- |
| Name | Value |
| Request force cycle counter ID | 0x04 |
| **Definition** | |
| #define KU8\_EOL\_REQUEST\_COUNTER\_LEVEL | |
| **Description** | |
| NA | |

### COUNTER\_LIMIT\_ENABLED

|  |  |
| --- | --- |
| Name | Value |
| Counter limit enable define | 0x00 |
| **Definition** | |
| #define COUNTER\_LIMIT\_ENABLED | |
| **Description** | |
| NA | |

### KU8\_NB\_MONITORED\_CYCLES

|  |  |
| --- | --- |
| Name | Value |
| Define used to limit the number of cycles monitored by the EOL module | 31 |
| **Definition** | |
| #define KU8\_NB\_MONITORED\_CYCLES | |
| **Description** | |
| NA | |

### KU8\_VARIABLE\_COUNTER\_LOW

|  |  |
| --- | --- |
| Name | Value |
| Variable counter low id | 1 |
| **Definition** | |
| #define KU8\_VARIABLE\_COUNTER\_LOW | |
| **Description** | |
| NA | |

### KU8\_VARIABLE\_COUNTER\_HIGH

|  |  |
| --- | --- |
| Name | Value |
| Variable counter high id | 2 |
| **Definition** | |
| #define KU8\_VARIABLE\_COUNTER\_HIGH | |
| **Description** | |
| NA | |

### KU8\_VARIABLE\_COUNTER\_MAX

|  |  |
| --- | --- |
| Name | Value |
| Variable counter max id | 4 |
| **Definition** | |
| #define KU8\_VARIABLE\_COUNTER\_MAX | |
| **Description** | |
| NA | |

### KU8\_VARIABLE\_COUNTER\_COMFORT

|  |  |
| --- | --- |
| Name | Value |
| Variable counter comfort id | 8 |
| **Definition** | |
| #define KU8\_VARIABLE\_COUNTER\_COMFORT | |
| **Description** | |
| NA | |

# EEPROM

The EEPROM parameters are all specified in [Doc3 = SBE\_4G\_NVP\_layout.xls].

Refer to this document for more details.

# Configuration

No special configuration for EOL software component.

# Compilation Options

No compilation options for EOL software component.